**HW BONUS REPORT**

**Osman Bahadır Erbek**

ProcessFile() metodum iki aşamadan oluşmaktadır.

**public** **void** processFile(String path) **throws** NumberFormatException, IOException {

FileReader fileReader = **new** FileReader(path);

String line;

BufferedReader br = **new** BufferedReader(fileReader);

**int** a =1;

**int** digits1[] = **new** **int**[2];

**while** ((line = br.readLine()) != **null**) {

String theDigits[] = line.split(" ");

**for**(**int** i = 0; i < theDigits.length; i++){

digits1[i] = Integer.*parseInt*(theDigits[i]);

}

numberofadd = digits1[0];

numberofdelete = digits1[1];

a--;

**if**(a==0) {

**break**;

}

}

br.close();

Bu dosya okuma işlemi dosyanın ilk satırındaki eklenecek öğrenci sayısı ve silinecek öğrenci sayısını bir değişken içerisinde atmaktadır. Sadece ilk satıdaki değerleri almak gerektiği için for döngüsünü “a” adındaki bir değişken aracılığıyla sadece bir kez çalıştırdım. Satırın ilk elemanı “numberoffadd” değişkenine ve ikinci elemanı “numberofdelete” değişkenin atandı.

String digits[][] = **new** String[100][100];

**int** actualLine = 0;

**int** aa=0;

**int** bb=0;

String line1 = buffer.readLine();

**while**(line != **null** && !line1.equals("")){

String theDigits[] = line.split(" ");

**for**(**int** i = 0; i < theDigits.length; i++){

aa=theDigits.length;

}

actualLine++;

bb=actualLine;

line = buffer.readLine();

}

String name="";

String surname="";

**int** idnumber=0;

**int** c=1;

**for**(**int** a1=0;a1<bb;a1++) {

**for**(**int** b=0;b<aa;b++) {

**if**(digits[a1][b]!= **null**) {

**if**(c==bb) {

**break**;

}

idnumber = Integer.*parseInt*(digits[c][0]);

name = digits[c][1];

surname = digits[c][2];

insertStudent( idnumber, name, surname, numberofadd);

numberofadd--;

c++;

}

}

}}

Bu arama fonksiyonunda ise dosya içerisinde bulunan öğrenci bilgilerini idnumber,name ve surname olmak üzere 3 farklı değişken içerisine atadım. İlk satır bilgilerim daha önce okunduğu için name = digits[c][1]; c değişkenini 1’den başlattım. Satır satır aldığım öğrenci bilgilerini “insertStudent” fonksiyonuna gönderdim. İnsertStudent fonksiyonu içerisine ek olarak “numberoffadd” değişkenini gönderdim. Bu ilk satırdan aldığım değişken her yeni öğrenci okunduğunda 1 eksilerek insertstudent içerisine gönderildi. İnsertStudent fonksiyonu içerisine gelen öğrenci bilgilerinin yanında eğer “numberofadd” değişkeni 0 veya 0dan küçükse tree’ye eklemedim.

**public** **void** insertStudent(**int** idnumber, String name, String surname,**int** i)

{

**if**(i<=0) {

**return**;

}

rootNode = insertStudent(idnumber, rootNode, name, surname);

}

Dosya içerisinde okuduğum öğrenci bilgileri öncelikli olarak bu fonksiyona gönderildi. Buraya ekstra olarak gelen numberofadd değişkeni 0’dan küçük ise tree’ye yerleştirilmedi. Buradan “node” bilgisiyle diğer insertStudent fonksiyonuna gönderildi.

**private** AVLNode insertStudent(**int** idnumber, AVLNode node, String name, String surname)

{

**if** (node == **null**)

node = **new** AVLNode(idnumber);

**else** **if** (idnumber < node.element)

{

node.left = insertStudent( idnumber, node.left, name, surname );

**if**( getHeight( node.left ) - getHeight( node.right ) == 2 )

**if**( idnumber < node.left.element )

node = rotateWithLeftChild( node );

**else**

node = doubleWithLeftChild( node );

}

**else** **if**( idnumber > node.element )

{

node.right = insertStudent( idnumber, node.right,name, surname );

**if**( getHeight( node.right ) - getHeight( node.left ) == 2 )

**if**( idnumber > node.right.element)

node = rotateWithRightChild( node );

**else**

node = doubleWithRightChild( node );

}

**else**

node.h = getMaxHeight( getHeight( node.left ), getHeight( node.right ) ) + 1;

**return** node;

}

Burada tree’ye yeni bir eleman eklerken avl ağaç yapısını korumak için bazı balance metotları kullandım. Bunları “javatpoint.com/” sitesinden aldığım bazı kodları modifiye ederek gerçekleştirdim. Recursive olarak çalışan bu fonksiyonda örneğin eklenecek idnumber o anki node’un id değerinden küçükse bir sol node yeriyle beraber kendisine tekrar gönderildi. Eğerki sol ve sağ tree yükseklikleri farkı 2’ye eşitse ve idnumber soldaki element’ten küçükse rotation işlemleri yapılmak üzere “rotatewithleftchild” fonksiyonuna gönderildi eğer büyükse iki kez rotation işlemi yapılacak olan doublewithrightchild metoduna gönderildi. Aynı işlemler idnumber node elemtinden büyük oldugu durumlar içinde yapıldı. Son olarak idnumber tree’ye eklendikten sonra node yüksekliği getMaxHeight fonksiyonu çağıralarak güncellendi.

**public** **void** printSearchPath(**int** element)

{

printSearchPath(rootNode, element);

**if**(ctrl==**false**) {

System.***out***.println("not found");

}

}

Aramak istenilen öğrenci öncelikli olarak bu fonksiyona gönderilir. Asıl arama yapılan fonksiyonda burdan gönderilerek tree içerisinde olup olmadığı “ctrl” boolean ifade ile kontrol edilir. Return olarak ctrl’yi döndüren asıl printsearchpath fonksiyonu ctrl’yi false olarak atarsa buradan not found uyarısı konsola gönderilir.

**private** **boolean** printSearchPath(AVLNode head, **int** element)

{

**while** ((head != **null**) )

{

**int** headElement = head.element;

**if** (element < headElement) {

head = head.left;

System.***out***.print(headElement +"-(left)->" ); }

**else** **if** (element > headElement) {

head = head.right;

System.***out***.print(headElement + "-(right)->"); }

**else** **if**( element == headElement)

{

ctrl=**true**;

System.***out***.print(headElement + "(Found)");

**break**;

}

**else** {

System.***out***.println("not found");

}

}

**return** ctrl;

}

}

Bu arama fonksiyonu aranın idnumber’ı sağa sola giderek tarar. Eğer ki idnumber o anki nood değerinden küçükse bir sola eğer büyükse bir sağa gider. Her yaptığı ilerleme sonrasında gittiği yönü consolda bastırır. Boolean olarak ctrl değişkenini döndüren bu fonksiyon, idnumber tree içerisinde varsa true olur eğer yoksa false olur. False olduğu taktirde bir yukardaki fonksiyonu “not found” şeklinde çalıştırır.

Reference:

1. https://www.javatpoint.com/